MONTGOMERY MODULAR MULTIPLIER AND METHOD THEREOF USING CARRY SAVE ADDITION

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] The present application claims priority from a Korean application having Application No. P2003-26482, filed 25 April 2003 in Korea, the disclosure of which is incorporated herein in its entirety by reference.

FIELD OF THE INVENTION

[0002] The present invention relates to the field of cryptosystems and, more particularly, to a Montgomery modular multiplier and method using carry save addition.

BACKGROUND OF THE INVENTION

[0003] For speed of computation of cryptosystems, fast exponential computation becomes important. One method used to accelerate computation is the Montgomery modular multiplication algorithm. The Montgomery modular multiplication algorithm provides a n-bit number:

[0004] $R=A^*B^*r^{-1} \bmod N$, (where the radix $r=2^n$) (1)

[0005] required in the modular exponential algorithm, where A, B, and N are the multiplicator, multiplicand, and modular number, respectively, and each has n bits.

[0006] A conventional hardware implementation of a Montgomery modular multiplication algorithm is shown in Figure 1, which utilizes a multiple modulus selector 1, a Booth Recoder 12, and an accumulator 2. The multiple modulus selector 1 selects a value for the multiple modulus (0, M, 2M, and 3M) and outputs the selected value to a carry propagation adder (CPA) 14. Obtaining a value of 3M requires an additional adder, increasing the hardware size and decreasing computational speed. CPA 14 is one of two carry propagation adders in the accumulator 2, the other is CPA 11. Each CPA added to the accumulator increases the overall propagation delay time and decreases computational speed. CPA 11 receives a partial product value from a multiplicand selector 13 and P[i], a previous value of the output of the accumulator 2. The multiplicand selector 13 receives the multiplicator and the output of the Booth Recoder 12 to obtain a partial product value (-2A, -A, 0, A, 2A). CPA 11 adds the partial product and P[i]. The output of CPA 11 is input to CPA 14 along with the value for the multiple modulus to obtain a resultant accumulation value for the i+1 iteration, P[i+1], obtaining a result for the Montgomery multiplication $P[i+1]=ABR^{-1}\text{mod } M$.

SUMMARY OF THE INVENTION

[0007] Exemplary embodiments of the present invention provide for methods of accelerating the speed of Montgomery modular multiplication and/or reducing power consumption by using a coding scheme which eliminates the need for an additional adder or memory when obtaining the multiple modulus value.

[0008] In exemplary embodiments of the present invention, a carry save adder (CSA) is used instead of a CPA in an accumulator to improve computation speed and propagation delay.

[0009] In exemplary embodiments of the present invention, a coding scheme eliminates the need for an adder or memory element for obtaining the multiple modulus value.

[0010] Further areas of applicability of embodiments of the present invention will become apparent from the detailed description provided hereinafter. It should be understood that the detailed description and specific examples, while indicating exemplary embodiments of the invention, are intended for purposes of illustration only and are not intended to limit the scope of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] Embodiments of present invention will become more fully understood from the detailed description and the accompanying drawings, wherein:

[0012] Figure 1 is an illustration of a background art hardware implementation of a Montgomery modular multiplication algorithm;

[0013] Figure 2 is an illustration of a modular multiplier of an exemplary embodiment of the present invention;

[0014] Figure 3 is a table describing selection criteria for the multiple of modulus MM_I in an exemplary embodiment of the present invention;

[0015] Figure 4 is a table describing selection criteria for the partial product PP_I in an exemplary embodiment of the present invention;

[0016] Figure 5 is an illustration of an accumulator of an exemplary embodiment of the present invention;

[0017] Figure 6 is an illustration of a complete compressor of an exemplary embodiment of the present invention;

[0018] Figure 7 is an illustration of a reduced compressor of an exemplary embodiment of the present invention; and

[0019] Figure 8 is an illustration of an accumulator of an exemplary embodiment of the present invention.

[0020] Figure 9 is an illustration of a configuration of a kth bit multiplexer of an exemplary embodiment of the present invention.

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS OF THE PRESENT INVENTION

[0021] The following description of exemplary embodiment(s) is merely illustrative in nature and is in no way intended to limit the invention, its application, or uses.

[0022] Figure 2 illustrates a modular multiplier 1000 of an exemplary embodiment of the present invention. The multiplier 1000 can include a modulus (M) stored in a register 200, a multiplicand (A) stored in a register 201, a multiplicator (B) stored in a register 202, a Booth recoder 210, a Modulus recoder 220, a multiplexer (MUX) 230 aiding in the computation of the multiple modulus MM_I, a MUX 240 aiding in the computation of the partial product PP_I, and an accumulator 250 for aiding in the computation of the modular multiplication. The accumulator 250 can input a partial product value PP_I, a multiple modulus value MM_I, and a compensating word signal (CW) and produce a result for the Montgomery multiplier. In exemplary embodiments of the present invention, the positive value M has n bits (M[n-1:0]). The positive or negative value A has n+1 bits (A[n:0]), one bit for a sign bit, and the multiplicator B has even bits. If n is even, B can have n+2 bits, two bits being sign bits. Or if n is odd, B can have n+1 bits, one bit being a sign bit.

[0023] In exemplary embodiments of the present invention, register 200 provides the modulus M and $\underline{M}$, where $\underline{M}$ is the one's complement of M. Similarly register 201 provides the multiplicand A and $\underline{A}$, where $\underline{A}$ is the one's complement of A and register 202 provides the multiplicator B.

[0024] The multiplier 1000 can solve for modular multiplication in an iterative process. The Modulus recoder 220 and the multiplexer 230 are used to select multiple modulus (MM_I) values. To select MM_I values, the Modulus recoder 220 receives iterative data from the accumulator 250. In an exemplary embodiment of the present invention the iterative data, $SPP_I[1:0]$, is based on the two LSBs of values in a sum ($S_I[1:0]$) and carry ($C_I[1:0]$) registry of the accumulator 250, two LSBs of the partial product value ($PP_I[1:0]$), and a partial product negation indicating signal NEG_PP. $C_I[1:0]$ and $S_I[1:0]$ can be combined in a two-bit adder 260 to form a combined signal. The combined signal can be combined with $PP_I[1:0]$ and NEG_PP in a two-bit adder 270 to form $SPP_I[1:0]$. In addition to $SPP_I[1:0]$ the Modulus recoder 220 inputs the second least significant bit of the Modulus, M[1]. The Modulus recoder 220 uses $SPP_I[1:0]$ and M[1] to generate output signals, which can determine the selection of a multiple modulus MM_I value. The discussion above, with respect to exemplary embodiments of the present invention, is not intended to limit the bit size of values. SPP_I can have more than two bits, as can other elements of the embodiment (e.g., adder 260 can be more or less than a two-bit adder).

[0025] The Modulus recoder 220 can output multiple signals (e.g., a multiple modulus selection signal SEL_MM[1:0], a multiple modulus negation indicating signal NEG_MM,...). In an exemplary embodiment of the present invention, the Modulus recoder outputs SEL_MM[1:0] to the multiplexer 230, which uses the value of SEL_MM[1:0] to select one of four possible values of MM_I (e.g., 2M, M, 0, M). The multiplexer (MUX) 230 inputs the modulus M and, in an exemplary embodiment, two LSBs of the multiple modulus selection signal SEL_MM[1:0], outputing the value of MM_I. MM_I is sent to the accumulator 250. The multiple modulus negation indicating signal NEG_MM can be combined in a half adder 47 with the partial product negation indicating signal NEG_PP to obtain the compensatory word signal CW. CW is sent to the accumulator 250.

[0026] NEG_MM is used to indicate whether the selected value of MM_I should be bit-inverted. Likewise NEG_PP is used to indicate whether the selected partial product PP_I should be bit-inverted. The PP_I value is based upon operations performed by the Booth recoder 210, the multiplexer 240 and an AND gate 280. PP_I is sent to the accumulator 250 along with MM_I and CW.

[0027] Although Figure 2 illustrates the use of 4:1 multiplexers (MUX), exemplary embodiments of the present invention are not limited to a particular ratio value of the multiplexer, nor is the accumulator limited to a 5-2 compressor. For example one 4-1 MUX can be replaced by three 2-1 MUXs.

[0028] Figure 3 illustrates a coding scheme in accordance with exemplary embodiments of the present invention. Although Figure 3 shows three inputs to the Modulus recoder 220, M[1] and SPP_I [1:0], the present invention can have a variety of inputs and outputs depending upon the design criteria. Typical values of the multiple modulus MM_I are (0, M, 2M, 3M). As described above the value 3M requires an additional adder or memory element to add 1M to 2M to obtain the value of 3M. An additional adder and/or memory element contributes to hardware size and/or computational delay, which affects computational speed and power usage. The coding scheme shown in Figure 3 utilizes bit-inversion and bit-shift to obtain the value of MM_I without an additional adder or memory element. The Modulus recoder 220 inputs M[1], the second least significant bit of the Modulus M, and, in an exemplary embodiment, SPP_I[1:0], two LSBs of SPP_I. Modulus recoder 220 outputs a modulus selection signal SEL_MM[1:0]. SEL_MM[1:0]is used to select one of four possible multiple modulus numbers (0, $\underline{M}$, M, 2M). The signal NEG_MM indicates whether a bit-inversion is used, obtaining $\underline{M}$. The resultant selected multiple modulus value MM_I is sent to the accumulator 250. The discussion above, with respect to exemplary embodiments of the present invention, is not intended to limit the bit size of values. SPP_I can have more than two bits as can other elements of the embodiment.

[0029] In another exemplary embodiment of the present invention, a similar method of decreased hardware size, increased computational speed and power reduction can be used with the Booth recoder 210 as shown in Figure 2 and 4. As mentioned above the multiplier 1000 solves for modular multiplication in an iterative process, which includes the supply of MM_I and partial product values (PP_I) to the accumulator 250. The Booth recoder 210 and multiplexer 240 are used to select partial product (PP_I) values (e.g. 0, A, 2A, $\underline{A}$, $\underline{2A}$) to supply to the accumulator 250. The Booth recoder 210 inputs the two LSBs of the multiplier (B[1] and B[0]) and B[r], a previous iteration's value of B[1] and outputs three signals, a partial product selection signal SEL_PP[1:0], a partial product enablement signal EN_PP, and a partial product negation indicating signal NEG_PP.

[0030] To select PP_I values, the Booth recoder 210 outputs the partial product selection signal SEL_PP[1:0] to the multiplexer 240 for selecting one of four possible values ($\underline{2A}$, $\underline{A}$, A, 2A). The multiplexer 240 receives the value of the multiplicand (A) and SEL_PP[1:0] and outputs a value to an AND gate 280. The AND gate 280 receives the input from the multiplexer 240 and a partial product enabling signal EN_PP from the Booth recoder 210. The AND gate 280 outputs the selected value of the partial product (PP_I) to the accumulator 250. When EN_PP has a zero value the AND gate 280 outputs a zero value for PP_I to the accumulator 250. The partial product negation signal NEG_PP is input to the half adder 47. A value of 1 for NEG_PP indicates that a bit-inversion should be performed on PP_I obtaining one of the values $\underline{2A}$ or $\underline{A}$ for a new PP_I value input to the accumulator 250.

[0031] In addition to PP_I and MM_I values, the compensating word signal (CW) is sent to the accumulator 250 from the half adder 47. The accumulator 250 inputs PP_I, MM_I, and CW into a combination of full and half compressors, which are used to add in a carry save adder (CSA) and propagate in a carry propagate adder (CPA). Conventional accumulators (Figure 1) use CPAs in each iteration, which as discussed above, results in accumulated propagation delay resulting in a low operating frequency. Use of a CSA reduces accumulated propagation delay increasing computation speed and decreasing computation power usage resulting in a high operating frequency.

Exemplary embodiments of the present invention utilize a combination of CSA and CPA to increase computation speed and decrease power usage. For example, in an exemplary embodiment of the present invention, only one CPA is used during the final iteration, while the previous iterations use a CSA. Figures 5 and 8 show two exemplary embodiments of the present invention and are for illustrated purposes only and not intended to limit the scope of the present invention to a particular configuration of use of CSA and a CPA.

[0032] An exemplary accumulator 500 in accordance with the present invention is illustrated in Figure 5. The accumulator is composed of n+2 series of 5-2 compressors, broken into full compressors (e.g., 520) and reduced compressors (e.g., 510), where n is the bit length of modulus value M. The accumulator 500 stores sum (S) and carry (C) values in a sum register 530 (S_REG) and a carry register 540 (C_REG), respectively. The outputs of the S_REG 530 and the C_REG 540 are input to a carry propagation adder 550, which converts a redundant number to a normal number, storing the value in a final register 560 (F_REG).

[0033] Input to the accumulator 500, in an exemplary embodiment of the present invention, are compensating word CW[1:0], the multiple modulus value MM_I and the partial product value PP_I. The first two full compressors, 570 and 520, input CW[1:0] along with MM_I[1:0] and PP_I[1:0]. The remaining reduced compressors 510, 580, 590, etc. use the remaining bits of the multiple modulus value MM_I[n+1:2], and the partial product value PP_I[n+1:2]. The last compressor 580 (n+2 compressor) prevents overflow and the first compressor 570 (n=0) is a full compressor missing a third full adder. Other exemplary embodiments can have a various number of bits for the various variable values (e.g., CW, MM_I, PP_I,...) and discussion herein should not be interpreted to limit the bit sizes of the variables.

[0034] Exemplary configurations of full 600 and reduced 700 compressors are shown in Figures 6 and 7, respectively. Each compressor is used to obtain a next value (I+1) using a current value (I) and other inputs. Figure 6 illustrates a full compressor 600 in accordance with an exemplary embodiment of the present invention. The full compressor 600 can have a plurality of inputs. In an exemplary embodiment a full compressor 600 can have five inputs, a current carry word bit value (C_I) obtained from a next carry word bit value from a compressor one bit higher, a current sum word bit value (S_I) obtained from a next sum word bit value from a compressor two bits higher, a compensating word value (CW), a partial product value (PP_I), and a multiple modulus value (MM_I). It is noted that inputted current carry word bit value have an index of "I" in the current compressor, whereas when leaving the higher bit compressor a value is output as the next carry word bit value $C_{I+1}[k+1]$, where k represents the current "kth" compressor or kth-bit compressor. The next carry word bit value $C_{I+1}[k+1]$ is input to the carry register 540, which outputs the current carry word bit value C_I to the kth compressor, as indicated above. The current sum word bit value $S_I[k]$ is likewise obtained by a next sum word bit value from the k+2 compressor $S_{I+1}[k+2]$ input to the sum register 530. The values are used by the full compressor 600 to obtain next carry word bit and next sum word bit values for the particular bit k, $C_{I+1}[k]$ and $S_{I+1}[k]$ respectively. These values are then passed to their respective carry and sum registers 540 and 530 (as shown in Figure 5). The outputs of the carry and sum registers, 540 and 530 respectively, serve as inputs to lower bit compressors as described above. The next carry word bit ($C_{I+1}[k]$) and next sum word bit ($S_{I+1}[k]$) values can be related by Equation (2).

[0035] $(2C_{I+1}[k]$
$$+2CO1[k]+2CO2[k]+S_{I+1}[k])=(C_I[k]+S_I[k])+PP_I[k]+MM_I[k]+CW[k]+CI1[k]+CI2[k] \qquad (2)$$

[0036] where if k>1, CW[k] is not an input and is effectively 0.

[0037] In an exemplary embodiment of the present invention the full compressor 600 is composed of three full adders. The first full adder 610 inputs the values C_I, S_I, and CW and outputs a first full adder carry (FCO1) and a first full adder sum (FSO1). FCO1 serves as a first output carry CO1, which can be a secondary first input CI1[k+1] for the next higher bit compressor (k+1). The second full adder 620 inputs FSO1, the partial product bit value PP_I[k] and the multiple modulus bit value MM_I[k] associated with the bit designation (k) of the compressor. The second full adder 620 outputs a second full adder carry (FCO2) and a second full adder sum (FSO2). FCO2 serves as a first output carry CO2, which can be a secondary second input CI2[k+1] for the next higher bit compressor (k+1). The third full adder 630 inputs FSO2, and CI1[k-1] and CI2[k-1] from a lower bit compressor (k-1). The third full adder 630 outputs a third full adder carry (FCO3) and a third full adder sum (FSO3). FCO3 serves as the next carry word bit value C_{I+1}, which is used to obtain the input C_I to a lower bit compressor (k-1). FSO3 serves as the next sum word S_{I+1}, which is used to obtain the input S_I to a two bits lower compressor (k-2). The first full compressor 570 corresponding to bit 0 does not output next carry or sum words, thus the third full adder is not needed. Likewise the second full compressor 520 corresponding to bit 1 does not output a next sum word bit value.

[0038] The compensating word CW[1:0] has two bits and thus requires two compressors, one for each bit. Thus, the first two compressors, 570 and 520, are full compressors inputting a plurality of values. In exemplary embodiments the full compressors 570 and 520 input five values. The higher bit compressors [2:n+2] input a plurality of values that are less than that input to compressors 570 and 520 and are referred to as reduced compressors 510. Reduced compressors replace the first full adder with a half adder. Thus, the half adder 710 in the reduced compressor inputs the values C_I and S_I and outputs a first half adder carry (HCO1) and a first half adder sum

(HSO1). HCO1 serves as a first output carry CO1, which can be a secondary first input CI1[k+1] for the next higher bit compressor (k+1). The second full adder 720 inputs HSO1, the partial product bit value PP_l[k]and the multiple modulus bit value MM_l[k] associated with the bit designation (k) of the compressor. The second full adder 720 outputs a second full adder carry (FCO2) and a second full adder sum (FSO2). FCO2 serves as a second output carry CO2, which can be a secondary second input CI2[k+1] for the next higher bit compressor (k+1). The third full adder 730 inputs FSO2, and CI1[k-1] and CI2[k-1] from a lower bit compressor (k-1). The third full adder 730 outputs a third full adder carry (FCO3) and a third full adder sum (FSO3). FCO3 serves as the next carry word bit C_{l+1}, which serves as input C_l to a lower bit compressor (k-1) after passing to the carry register 540. FSO3 serves as the next sum word bit S_{l+1}, which serves as input S_l to a two bits lower compressor (k-2) after passing to the sum register 530.

[0039] The accumulator 500 of Figure 5, in accordance with an exemplary embodiment of the present invention, links in series full compressors and reduced compressors, the number of which depends on the input bit size of the multiple modulus value (MM_l) and the partial product value (PP_l). The two LSB compressors are full compressors that use the compensating word (CW) as an input. The first bit compressor 570 outputs CO1[0] and CO2[0], which become secondary inputs to the next higher bit (second bit) compressor 520, CI1[1] and CI2[1] respectively. This continues until the highest bit compressor (n+2), which does not output carry outputs(CO1[n+2] and CO2[n+2]). The highest bit compressor prevents overflow and its secondary inputs are obtained from its own next carry word bit and next sum word bit values.

[0040] Each compressor's next carry word bit value and next sum word bit value are passed to their respective carry and sum registers 540 and 530, respectively.

The final results are generated in a separated form (redundant number) one part stored in the sum register 530 and the other part stored in the carry register 540. To obtain the final single word result $S_N[n:0]$ the value stored in the sum register 530 and the value stored in the carry register 540 are added in a carry propagation adder (CPA) 550, and the final single word result $S_N[n:0]$ is stored in a final register (F_REG) 560. The use of the CSA mode instead of a pure CPA mode of the conventional systems is that, for example in the exemplary system describe in Figure 5, the CSA compressors have three delay paths, one associated with each adder. In a conventional accumulator, a delay path exists for each bit.

[0041] Thus, for the exemplary embodiment of the present invention shown in Figure 5, three delay paths exist for all of the compressors combined, regardless of the bit size n, since they are configured using carry save addition. In a conventional system, there would be “n” delay paths. Thus, the exemplary configuration can significantly improve the computational speed of a modular multiplication. For example, in a 1024 bit multiplier a conventional system will have an accumulator with 1024 delay (full adder paths) whereas exemplary embodiments of the present invention would have only the path delays associated with a single full compressor or reduced compressor, e.g., 3. Thus, in this example, a multiplier based on an exemplary embodiment of Figure 5 would be 300 times faster than the conventional system. In an exemplary embodiment of the present invention shown in Figure 5, a CPA is used only once.

[0042] Other exemplary embodiments of the present invention include a variety of combinations of switching between CSA and CPA modes in the accumulator. For example, Figure 8 illustrates an accumulator 800 according to an exemplary embodiment of the present invention, where multiplexers MXG_{n+1} to MXG_0 are used in combination with the compressors to switch between CPA and CSA mode when desired. Such a configuration no longer has a CPA 550 to convert a redundant number to a normal number. The accumulator 800 shown in Figure 8 is selectively worked in the CSA or CPA mode, thus the output is already in normal number format. Removing the CPA 550 reduces the size of the hardware needed.

[0043] The multiplexers (MXG_{n+1} to MXG_0) can control the electrical connections between full adders in the compressors. As shown in Figure 8, the first two bit compressors 870 and 820 are analogous to the description and operation of the compressors 520 and 570, respectively, except that the next carry word bit value ($C_{l+1}[k]$) is not only passed to the carry register 840 to obtain a current carry word bit value $C_l[k-1]$, used by the lower bit compressor 870, $C_{l+1}[k]$ is passed to the next higher bit compressor [k+1] as input to a multiplexer associated with the higher bit compressor MXG_{k-2}

[0044] Figure 9 illustrates a configuration of a kth bit multiplexer 900 in accordance with exemplary embodiments of the present invention. The computation mode (using CSA or using CPA) can be controlled by a switching signal (SW) 910. In an exemplary embodiment of the present invention, the kth bit multiplexer 900 can be placed between the second adder 720 and the third adder 730 of the reduced compressor 700 of Figure 7. Thus, the first input 901 into the first element 920 of the multiplexer 900 is FSO2 from adder 720, described above. The second input 902 to the first element 920 is the current carry word bit value ($C_l[k-1]$) from the k-1 bit compressor, where the current carry word bit value is obtained from the next carry word bit value for the k-1 bit compressor, $C_{l+1}[k-1]$, that has been passed to the carry register 840. The second element 930 of the multiplexer 900 inputs two values, the first 903 is the first output carry value, CO1[k-1], from the k-1 bit compressor (also the first secondary input to the kth bit compressor, CI1[k]), and the second 904 is the current sum word bit value $S_l[k]$ of the kth bit compressor, where the current sum word bit value is obtained by passing the next sum word bit value $S_{l+1}[k]$ to the sum register 830. The third element

940 of the multiplexer 900 also inputs two values, the first 905 is the second output carry value , CO2[k-1], from the k-1 bit compressor (also the second secondary input to the kth bit compressor, CI2[k]), and the second 906 is the next carry word bit value for the k-1 bit compressor, C_{I+1}[k-1].

[0045] The switching signal, SW 910, determines which of the two input values to each element 920, 930, and 940 pass to the third full adder 730. Depending on which values are passed determines which mode of operation occurred, a carry save addition or a carry propagation addition. If the value of SW 910 is zero then the compressors are operated in carry save addition mode. If the value is one then the bottom full adders of the compressors are connected in series and operated in carry propagation addition mode. The full adder 730 outputs a next carry word bit value and a next sum word bit value as described above. The exemplary embodiment described above uses two inputs per element 920, 930, and 940. The present invention is not limited to a particular number of inputs and other exemplary embodiments in accordance with the present invention have a plurality of inputs and a plurality of elements and multipliers.

[0046] Carry and sum words are computed during N iterations, where N is (n+2)/2 if n is even or (n+1)/2 if n is odd. Carry and sum values outputted in a current iteration cycle are added with those of a previous iteration cycle and stored in the carry register 840 (C_REG) and the sum register 830 (S_REG). The final result S_N[n:0] is obtained by adding carry and sum in the registers 830 and 840 respectively by varying the desired switching value SW 910.

[0047] The exemplary embodiment shown in Figure 8 allows a reduction in the hardware size since multiplexers may have much smaller size than the CPA adder 550 plus the F_REG 560.

[0048] The description of the invention is merely exemplary in nature and, thus, variations that do not depart from the gist of the invention are intended to be within the scope of the embodiments of the present invention. Such variations are not to be regarded as a departure from the spirit and scope of the present invention. For example multiplexers 230 and 240 can have a variety of ratio values. Likewise the multiplexers used in exemplary embodiments of the present invention, for example as shown in Figure 8 can be composed of a single multiplexer or individual multiplexers with varying inputs. Likewise the controlling signal can be switched so that a value of zero signifies the use of the CPA mode as opposed to the CSA mode and vice versa. Further variations of the exemplary embodiments of the present invention described herein will become apparent to one of ordinary skill in the art, such variations are intended to lie within the scope of the present invention.